John Mansfield, OSB No. 055390
MansfieldLaw
121 SW Morrison Ave., Suite 400
Portland, OR 97204
Telephone: 971.271.8615
john@mansfieldlaw.net

Jonathan Baker (appearing *pro hac vice*)
FARNEY DANIELS PC
411 Borel Avenue, Suite 350
San Mateo, California 94402
Telephone: (424) 278-5200
jbaker@farneydaniels.com

[Additional Counsel listed on signature page]

Attorneys for Plaintiff Memory Integrity, LLC

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF OREGON
PORTLAND DIVISION**

MEMORY INTEGRITY, LLC, Plaintiff, v. INTEL CORPORATION, Defendant.	Case No. 3:15-cv-00262-SI **DECLARATION OF MARK JONES, Ph.D. REGARDING CLAIM CONSTRUCTION**

I, Mark Jones, Ph.D., hereby declare as follows:

1. My name is Dr. Mark Jones. I submit this declaration in support of Plaintiff Memory Integrity, LLC. I have been asked to offer technical opinions relating to the claim construction of certain terms of U.S. Patent Nos. U.S. Patent Nos. 7,296,121, 7,103,636, 7,107,409, 8,572,206 and 8,898,254 ("the '121 patent," "the '636 patent," "the '409 patent," "the '206 patent," and "the '254 patent," collectively "the patents-in-suit").

2. I am a Professor of Electrical and Computer Engineering at Virginia Tech in Blacksburg Virginia. I graduated summa cum laude from Clemson University in 1986 with a B.S. in Computer Science and a minor in Computer Engineering while holding a National Merit Scholarship and the R. F. Poole Scholarship. I then graduated from Duke University in 1990 with a PhD in Computer Science while holding the Von Neumann Fellowship.

3. Upon graduation, I joined the Department of Energy at their Argonne National Laboratory facility. My responsibilities there included the design and use of software for computers with hundreds of processing elements. This software was designed for compatibility with new parallel computer architectures as they became available as well as with other large software components being written in the Department of Energy. While with DOE, I received the IEEE Gordon Bell Prize.

4. In 1994, I joined the Computer Science faculty at the University of Tennessee. My teaching responsibilities included Computer Architecture and Computer Networking. My research interests included the design and use of software that used the collective power of large groups of workstations. While at the University of Tennessee, I received a CAREER Award from the National Science Foundation.

5. In 1997, I joined the Electrical and Computer Engineering faculty at Virginia Tech. My teaching responsibilities have included computer organization, computer architecture,

a variety of programming courses, and parallel computing. I have been cited multiple times on the College of Engineering's Dean's List for teaching. I have attached a true and correct copy of my curriculum vitae as Exhibit A, which further sets forth my qualifications.

6. I have no financial interest in either party or in the outcome of this proceeding. I am being paid for my work as an expert in these matters on an hourly basis. My compensation is not dependent on the outcome of these proceedings or the content of my opinions. My opinions, as explained below, are based on my education, experience, and background in the fields discussed above, and my review of the materials cited by the parties in the Joint Claim Construction chart, petitioners' submissions in the pending IPRs of the '121 Patents, and the other materials discussed herein.

7. It is my opinion that that one of ordinary skill in the field of the patents-in-suit—cache coherent microprocessors—as of November 4, 2002, would have at least a bachelor's degree in electrical engineering, computer engineering, or computer science, and at least two years of experience in the design of multiprocessor systems.

A. "point-to-point architecture" (each patent-in-suit)

8. The term "point-to-point architecture" is recited in claims 15-18, 21, 25-26, and 34 of the '636 Patent, claims 1-3, 6-12, 18-20, 22-23, 25-30, 34-38, 42-43, 45-49, and 51-52 of the '409 Patent, claims 1-6, 8, 11-17, and 19-25 of the '121 Patent, claims 1-4, 7, 12, 14, 15, 19-22, 24-29, 31, and 32 of the '206 Patent, and claims 2, 3, and 5-7 of the '254 Patent. I understand that Memory Integrity has proposed that the term be construed as "an architecture in which multiple processors or processing nodes are directly connected to each other through point-to-point links and share the same memory address space". Joint Claim Chart, Ex. B at 3. I also understand that Intel has proposed the term be construed as "an architecture in which multiple processors are directly connected to each other through point-to-point links."

9. There are two primary differences between the parties' constructions: (1) MI's construction indicates that point-to-point architectures can connect either processing nodes or processors, and (2) MI's construction requires that the processors or processing nodes share the same memory space. It is my opinion that the construction proposed my Memory Integrity is how one of ordinary skill in the art, informed by the specification and teachings of the patent, would interpret the phrase "point-to-point architecture" in the claims of the patents.

10. As to the first difference between the parties' constructions, only the '636 and '409 patents discuss "processors" being interconnected by a "point-to-point architecture." '636 Pat. cls. 15 ("the first plurality of processors and the first cache coherence controller interconnected in a point-to-point architecture . . . the second plurality of processors and the second cache coherence controller interconnected in a point-to-point architecture"), 21 ("the first plurality of processors and the first cache coherence controller interconnected in a point-to-point architecture . . . the second plurality of processors and the second cache coherence controller interconnected in a point-to-point architecture"), 25 ("wherein the request cluster includes a plurality of processors interconnected in a point-to-point architecture"), 26 ("wherein the remote cluster includes a plurality of processors interconnected in a point-to-point architecture."); '409 Pat. cls. 1 ("the first plurality of processors and the first cache coherence controller interconnected in a point-to-point architecture; . . . the second plurality of processors and the second cache coherence controller interconnected in a point-to-point architecture."), 6 ("the first plurality of processors and the first cache coherence controller interconnected in a point-to-point architecture; . . . the second plurality of processors and the second cache coherence controller interconnected in a point-to-point architecture"), 7 ("the plurality of local processors are arranged in a point-to-point architecture"), 25 ("a local cluster of processors connected through a point-to-point architecture . . . a remote cluster of processors connected through a point-to-point

architecture"), 34 ("a local cluster of processors connected through a point-to-point architecture . . . a remote cluster of processors connected through a point-to-point architecture"), 42 ("a first cluster having a first processor and a first controller interconnected in a point-to-point architecture"), 51 ("a first cluster of processors interconnected with a cache coherence controller in a point-to-point architecture"), and 52 ("a first cluster of processors interconnected with a cache coherence controller in a point-to-point architecture").

11. By contrast, the claims of the '121, '206, and '254 Patents do not recite "processors" interconnected by a point-to-point architecture and instead mention "nodes" or "processing nodes" interconnected by a point-to-point architecture. '121 Pat. cls. 1 ("plurality of processing nodes interconnected by a first point-to-point architecture"), 16 ("a plurality of processing nodes interconnected by a first point-to-point architecture"), 25 ("a plurality of processing nodes interconnected by a first point-to-point architecture") ; '206 Pat. cls. 1 ("a plurality of local nodes and an interconnection controller interconnected by a local point-to-point architecture"), 21 ("each cluster including a plurality of local nodes and an instance of the interconnection controller interconnected by a local point-to-point architecture"), 30 ("the processing nodes and interconnection controller are interconnected in a point-to-point architecture"); '254 Pat. cl. 2 ("wherein the processing nodes and interconnection controller are interconnected in a point-to-point architecture").

12. The '121, '206 and '254 Patents teach that the term "node" or "processing node" are not synonymous with the term "processor." For example, the '121 Patent teaches that:

> It should further be noted that the terms node and processor are often used interchangeably herein. However, it should be understood that according to various implementations, a node (e.g., processors 202 a-202 d) may comprise multiple sub-units, e.g., CPUs, memory controllers, I/O bridges, etc.

'121 Pat. at 6:48-57.

13. Similarly, the specification of the ’206 Patent, which is substantially identical to the specification of the ’254 Patent, teaches that “Each cluster includes a plurality of local nodes and an interconnection controller interconnected by a local point-to-point architecture.” ’206 Pat. at Abstract. The’206 Patent’s specification also teaches that:

> Although generally a node may correspond to one or a plurality of resources (including, for example, a processor), it should be noted that the terms node and processor are often used interchangeably herein. According to a particular implementation, a node comprises multiple sub-units, e.g., CPUs, memory controllers, I/O bridges, etc., each of which has a unit ID.

’206 Pat. at 6:36-42; *see also* ’206 Pat. at 6:9-10 (“According to a specific embodiment, one of the nodes in each multi-processor cluster is an interconnection controller”). The ’206 Patent’s specification also teaches that, according to certain preferred embodiments, components on the point-to-point network can be identified using a “node ID, “unit ID,” and “cluster ID.” ’206 Pat. at Fig. 12, 5:23-30; 9:24-52.

14. Thus, the ’121, ’206 and ’254 Patents teach that the term “node” *can* be simply a single processor, but the term “node” is not limited to a single processor, and may include, for example, multiple “CPUs” (i.e. multiple processors) as well as additional sub-units, including “I/O bridges.” This is consistent with the ordinary meaning of “node” in the field.

15. Thus, Intel’s construction of “point-to-point architecture” would unduly limit the claims of the patents-in-suit directed to “processing nodes” interconnected by a “point-to-point architecture” to only single processors, which is narrower than the full scope of a “node” or “processing node” in the patents or in the field.

16. I understand that Intel cites to a passage in the ’636, ’409, and ’121 Patents stating that “In a point-to-point architecture, a cluster of processors includes multiple processors directly connected to each other through point-to-point links.” ’636 Pat. at 5:8-11; ’409 Pat. at 4:32-35; ’121 Pat. at 4:38-40. Notably, this passage does not appear in the ’206 or ’254 Patents.

Moreover, in context, and in consideration of the statement in the ’121 Patent that “the terms node and processors are often used interchangeably herein,” I do not believe that the passage cited by Intel is intended to narrow the ordinary meaning of point-to-point architectures to only interconnecting “processors” instead of “processing nodes,” as claimed by the ’121 Patent. In particular, this passage purports to describe the structure of a “point-to-point architecture” within a “cluster,” but only claims 5 and 6 of the ’121 Patent recite a “cluster”—most of the claims of the ’121 Patent do not require a “cluster.”

17. As to the second difference between the parties’ constructions, the patents make clear that, in the context of the claimed inventions of the patents-in-suit, the point-to-point architecture of the claims connects processors or processing nodes that share the same memory address space. For example, the ’121,’636, and ’409 patents teach that the use of a shared memory address space is an important distinction between a point-to-point architecture and that of another method of interconnection—external networks. In particular, the ’121 Patent states that “By using a single address space, internal point-to-point links can be used to significantly improve intercluster communication over traditional external network based multiple cluster systems.” ’121 Pat. at 5:19-39. Similarly, the ’409 and ’636 Patents teach that “By using point-to-point links instead of a conventional shared bus or external network, multiple processors are used efficiently in a system sharing the same memory space.” ’636 Pat. at 5:11-14, 6:6-10; ’409 Pat at 4:35-38, 5:28-32. Additionally, each of the ’121, ’409, and ’636 Patents teach that “delay in intercluster transactions in an architecture using a shared memory space is significantly less than the delay in conventional message passing environments using external networks such as Ethernet or Token Ring.” ’121 Pat. at 5:19-39; ’636 Pat. at 6:6-10; ’409 Pat. at 5:28-32.

18. “External networks such as Ethernet or Token Ring” refer to methods for communicating between separate computers over a network. In such systems, each computer

typically has its own memory and memory address space. Those memory addresses cannot be used, without modification, for distributed computing tasks because each system independently manages its memory address space and, therefore, a particular memory address on one computer would not store the same information (e.g., a variable) at that same memory address on another computer. In contrast, the patents distinguish the claimed point-to-point architecture from such "external networks" because the point-to-point architecture connects processors or nodes within a computer system in which a shared memory space is used.

19. Moreover, the common specification shared by the '206 and '254 patents describes each of the embodiments as using a shared memory address space, which can be implemented due to the point-to-point architecture and the claimed interconnection controller. '206 Pat. at 2:42-44 ("FIG. 11 is an exemplary mapping of protocol engines in a processor cluster to a global memory space in a multi-cluster system"); 3:15-16 ("the multiple processors in the multiple cluster architecture shown in FIG. 1A share a global memory space"), 6:11-13 ("an interconnection controller, e.g., interconnection controller 230 of FIG. 2, which manages the hierarchical mapping of information thereby enabling multiple clusters to share a single memory address space"), 6:23-24 ("each node in a cluster has a portion of a shared memory space with which it is associated."); 7:14-16 ("the global memory space is shared by 4 such clusters").

20. Thus, because the specifications of the patents-in-suit consistently and repeatedly describes the processors or processing nodes being interconnected in a shared memory address space, one of skill in the art would understand the "point-to-point architecture" of the claims of the patents-in-suits as connecting processing nodes sharing a common memory space.

B. "a cache access request" ('409 and '636 Patents)

21. The term "a cache access request" is recited in claims 11-18, 21-31, and 33-36 of the '636 Patent and claims 1-3, 6-12, 18-20, 22-23, 25-30, 34-38, 45-46, and 51-52 of the '409

Patent. I understand that Memory Integrity has proposed that the term be construed as "a request for access to data that may be stored in cache." Joint Claim Chart, Ex. B at 3. I also understand that Intel has proposed the term be construed as "a request for access to data stored in cache." Joint Claim Chart, Ex. B at 3.

22. It is my opinion that the construction proposed my Memory Integrity is how one of ordinary skill in the art, informed by the specification and teachings of the patent, would interpret the phrase "a cache access request" in the claims of the '636 and '409 Patents. It is also my opinion that Intel's proposed construction of the term as "a request for access to data stored in cache" is inconsistent with how one of skill in the art would interpret the phrase. In particular, Intel's construction is inconsistent with the normal operation of caching systems, and, in particular, appears to exclude what are called "cache misses." A brief explanation of the fundamental of caches in computer systems illustrates the fault in Intel's construction. The following explanation discusses the general theory of the operation of caches in a simple computer system and is not intended to precisely describe any particular cache system.[1]

23. Computers, in their operation, utilize a "hierarchy" for storing data. Hennessy & D. Patterson, *Computer Organization and Design: the Hardware/Software Interface* (2d ed. 1998) at 540-544. In the memory hierarchy, there are various types of memory that are implemented with different technology. At the top of the memory hierarchy is the fastest and smallest memory (e.g., registers on a processor) while the lowest level of the hierarchy is typically the largest and slowest memory (e.g., magnetic hard disk drives). Patterson at 109-110,

[1] In particular, this discussion focuses on a simplified example of a single processor system. Although all of these basic notions also apply to a system with multiple processors, there are additional steps and details that have to be addressed in such systems and which are not discussed below.

540-544, B-25. In the middle of the memory hierarchy is often a memory constructed using DRAM (dynamic random access memory) chips. Patterson at 540-544.

24. Traditionally, a "cache" in a computer is a form of temporary storage intended to improve the performance of computers and is interposed between a processor and main memory.[2] Patterson at 545-546. The cache is typically faster and smaller than main memory, but is slower and has more storage than the processor's file of registers. In normal operation, a cache is accessed when a processor attempts to read from a portion of memory.[3] Patterson at 545-546. If the requested block of data is stored in cache, the cache will return the requested data to the processor, obviating the need to wait for the slower main memory. Patterson at 545-546. However, because the cache is smaller than main memory, not all data can be stored in the cache. Thus, sometimes the processor will attempt to request a particular data block, but it will not be located in the cache. Patterson at 542-546. This is called a cache "miss." On the other hand, when the data block is located in the cache, this is called a cache "hit." When a cache miss occurs, the data block must be obtained from main memory.[4] Patterson at 542-546. In many implementations, this causes the cache to then start storing that the memory block, so it can be more quickly accessed in subsequent operations. Patterson at 545-546. However, because

[2] Note that other types of "caches" are known to one of ordinary skill, but this section describes the type of cache discussed in the patents-in-suit.

[3] Caches are often involved in memory write operations as well, but many of the details will depend on the particular implementation. Thus, for simplicity, this discussion focuses on memory reads.

[4] In many systems, there will be a hierarchy of different caches, with smaller and faster caches which are accessed first, and then one or more slower caches that are accessed if the faster caches "miss." In such systems, the main memory is not typically accessed until the last, slowest cache is missed.

caches, by their nature, are smaller than main memory, and because which data blocks will be accessed is not completely predictable, it is impossible to eliminate cache misses.[5]

25. Thus, Intel's construction is inappropriate because it excludes cache access requests that result in a cache miss. It makes no sense to define a "cache access request" in a manner that excludes misses. Moreover, in a typical system, there is no way to know whether or not a cache access request will miss until the cache is actually checked. Thus, Intel's definition is inconsistent with how caches work. On the other hand, the inclusion of the word "may" in Memory Integrity's construction accurately reflects that a data block which is the subject of a cache access request *may* be stored in the cache (i.e. a cache "hit") but it also *may not* be stored in the cache (i.e. a cache "miss") and there typically is no way to know if it will be a hit or a miss until the request is actually processed by the cache.

26. Nothing in the '409 or '636 Patents supports Intel's exclusion of cache misses from cache access requests. Although those patents do not expressly discuss cache misses, that is because they are concerned with more advanced cache techniques and are written from the perspective of one skilled in the art, who would already understand the fundamentals of how caches work, including cache misses. For example, the '409 and '636 patents teach that, consistent with the ordinary meaning of cache in the field, the purpose of the cache is for storing "frequently used data." *See, e.g.*, '636 Pat. at 1:46-65, '409 Pat. at 1:27-47. Thus, because only a subset of data is stored in the cache, this statement implies that some of the data that a processor may request would be absent from the cache and therefore miss. Additionally, the '409 and '636 patents state that "a local process or [sic] attempting to read the cache data block

[5] There are techniques which, in a multi-processor system, reduce the need to check some processors' caches in connection with some operations. Some of those techniques involve storing information about which processors have a particular data block in their cache. However, nothing in the claims of the '636 or '409 patents indicates that those patents are limited to systems employing such techniques.

can be allowed to access the data block without sending the requests through a serialization point in certain circumstances" and "[i]n one example, read access can be permitted when the cache block is valid and the associated memory line is not locked." '636 Pat. at 6:28-34, '409 Pat. at 5:50-56. This implies that, in the '636 and '409 Patents, cache access requests that are attempted may fail due to a data block being absent or in an invalid state.

27. Additionally, other relevant uses of the term "cache access request" in the field reflect that such requests can include misses. For example., in U.S. Patent Appl. No. 2002/0065992 A1, entitled "Software controlled cache configuration based on average miss rate," (published May 30, 2002), the abstract describes that, in the described system, "Data is loaded into various lines in the cache in response to cache access requests when a given cache access request misses." Thus, this clearly teaches that the understanding of the term "cache access request" in the field includes requests for which the cache does not contain the requested memory location (i.e. a cache miss).

28. Additionally, U.S. Patent No. 4,775,955 to Liu, entitled "Cache coherence mechanism based on locking," (issued Oct. 30, 1988) mentions "Upon a cache miss for a cache access request from IE 103 with 'content(CDIRD)' ≠CDID0, decisions are made to replace the LRU line in the selected congruence class." U.S. Pat. 4,775,955 at 9:52-55. Thus, this also clearly teaches that the term "cache access request" in the field includes requests for which the cache does not contain the requested memory location (a cache miss).

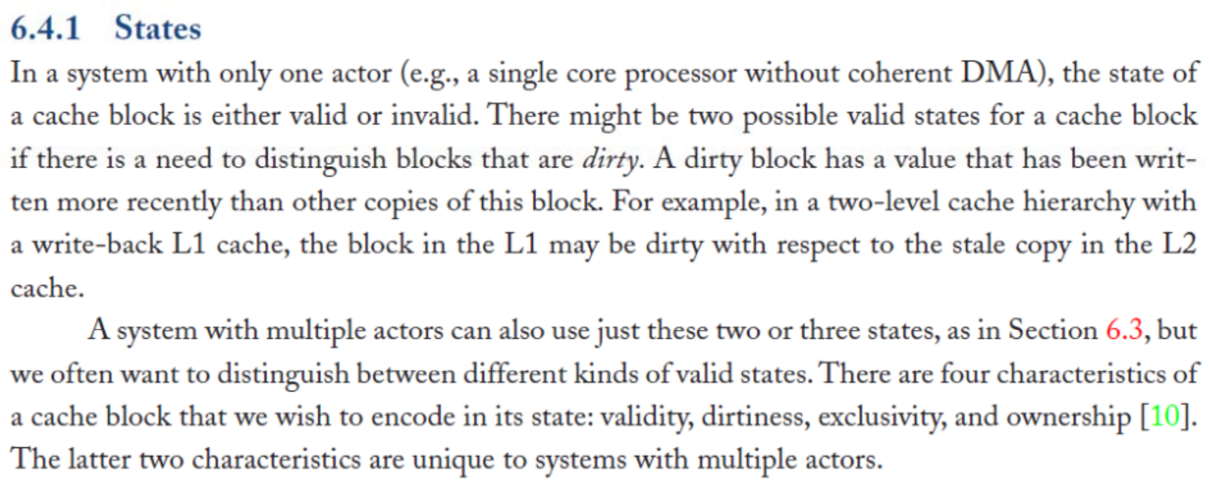
C. "states associated with selected ones of the cache memories" ('121 Patent)

29. The term "states associated with selected ones of the cache memories" is recited in each independent claim (claims 1, 16, and 25) of the '121 Patent. I understand that Memory Integrity has proposed that the term be construed as "cache coherence protocol states associated with selected ones of the cache memories". Joint Claim Chart, Ex. B at 7. I also understand that

Intel has proposed the term be construed as "Plain meaning. To the extent a construction is necessary: 'status of data stored in selected ones of the cache memories.'" Joint Claim Chart, Ex. B at 7.

30. It is my opinion that the construction proposed my Memory Integrity is how one of ordinary skill in the art, informed by the specification and teachings of the patent, would interpret the phrase "states associated with selected ones of the cache memories" in the claims of the '121 Patent. It is also my opinion that Intel's alternative proposed construction of the term as "status of data stored in selected ones of the cache memories" is inconsistent with how one of skill in the art would interpret the phrase, as it would divorce the term "state" from its well understood meaning in the field of cache coherency and the teachings of the specification of the '121 Patent.

31. Although the term "state" may have many broad and different meanings in general English usage, as well as in the specific field of computers, the term "state" connotes a specific meaning in the context of the field of cache coherency. In the field of cache coherency, the term "state" is understood to refer to a "cache coherence protocol state." For example, in Sorin et al., *A Primer on Memory Consistency and Cache Coherence* (2011), the author equates the term "state" with cache coherence protocol states. The table of contents of the Sorin book simply lists "States" as a heading, even though the section is dedicated to talking about cache coherence protocol states. Sorin 2011 at xi. Additionally, the "states" section of the Sorin book immediately begins discussing states, equating "states" with the states of a cache coherence protocol:



6.4.1 States

In a system with only one actor (e.g., a single core processor without coherent DMA), the state of a cache block is either valid or invalid. There might be two possible valid states for a cache block if there is a need to distinguish blocks that are *dirty*. A dirty block has a value that has been written more recently than other copies of this block. For example, in a two-level cache hierarchy with a write-back L1 cache, the block in the L1 may be dirty with respect to the stale copy in the L2 cache.

A system with multiple actors can also use just these two or three states, as in Section 6.3, but we often want to distinguish between different kinds of valid states. There are four characteristics of a cache block that we wish to encode in its state: validity, dirtiness, exclusivity, and ownership [10]. The latter two characteristics are unique to systems with multiple actors.

Sorin 2011 at 88. The Sorin book also teaches that "Many coherence protocols use a subset of the classic five state MOESI model first introduced by Sweazey and Smith" and that "The MOESI states, although quite common, are not an exhaustive set of stable states. . . . There are many possible coherence states, but we focus our attention in this primer on the well-known MOESI states." Sorin 2011 at 89-91. Again, this language reflects that, in the field of cache coherency, "states" are understood as referring to "cache coherence protocol states."

32. Moreover, equating "states" with "cache coherence protocol states" is not new—it was also the usage of the term "states" in the field of cache coherency at the time of the filing of the '121 Patent. For example, "Specifying and Verifying a Broadcast and a Multicast Snooping Cache Coherence Protocol," by Sorin et al. (2002), states that "A processor's access to a cache block is determined by the state of that block in its cache, and this state is generally one of the five MOESI (Modified, Owned, Exclusive, Shared, Invalid) states." Sorin 2002 at 1.

33. Additionally, the teachings of the '121 Patent both confirm that it is firmly embedded in the specific field of cache coherency, and that it also uses the term "state" to refer to cache coherence protocol states. The '121 Patent confirms that it is directed to the specific field of cache coherency from the very beginning of the Summary of the Invention section, stating that:

> Data access in multiple processor systems can raise issues relating to cache coherency. Conventional multiple processor computer systems have processors coupled to a system memory through a shared bus. In order to optimize access to data in the system memory, individual processors are typically designed to work with cache memory. In one example, each processor has a cache that is loaded with data that the processor frequently accesses. The cache is read or written by a processor. However, cache coherency problems arise because multiple copies of the same data can co-exist in systems having multiple processors and multiple cache memories.

'121 Patent at 1:26-38. Moreover, the '121 Patent describes the primary problem to be solved as "to provide techniques for improving data access and cache coherency in systems having multiple processors connected using point-to-point links." '121 Pat. at 2:39-42.

34. The '121 Patent's usage of the term "state" is consistent with "states" meaning cache coherence protocol states. For example, the '121 Patent describes that:

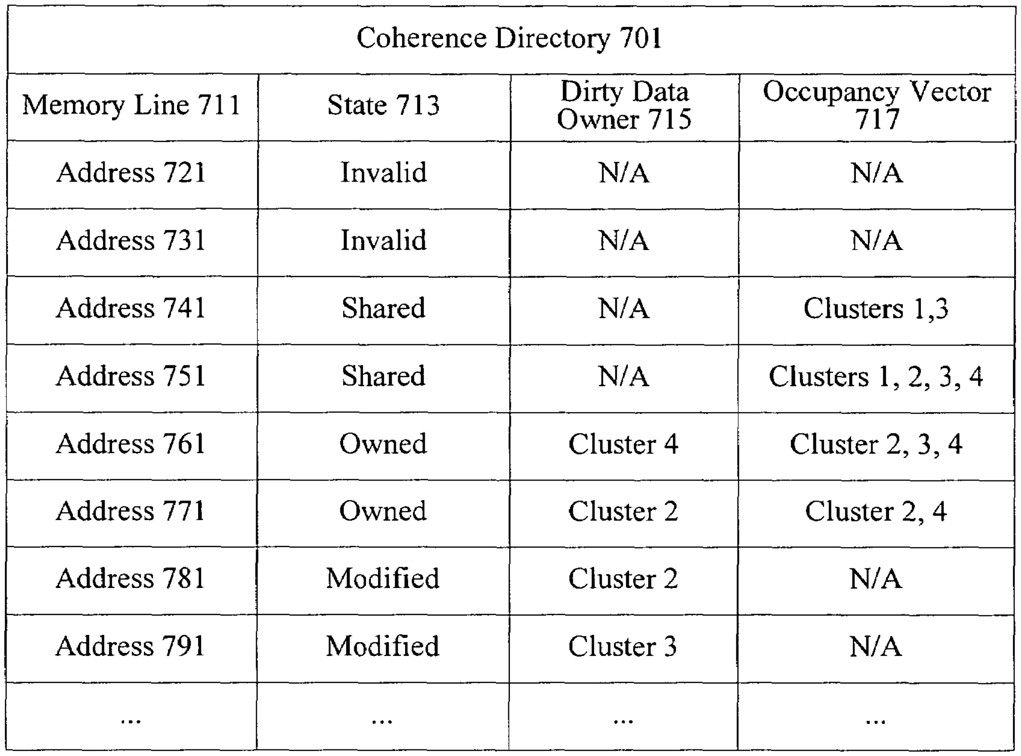
> [A] coherence protocol can contain several types of messages. In one example, a coherence protocol includes four types of messages; data or cache access requests, probes, responses or probe responses, and data packets. Data or cache access requests usually target the home node memory controller. Probes are used to query each cache in the system. The probe packet can carry information that allows the caches to properly transition the cache state for a specified line.

'121 Pat. at 9:21-29 (emphasis added). This reflects that the "state" discussed in the '121 Patent reflects the "state" of a memory line in a "[cache] coherence protocol."

35. Similarly, the '121 Patent explains that "[b]y using a ***coherence*** directory, global ***memory line state information*** (with respect to each cluster) can be maintained and accessed by a memory controller or a cache coherence controller in a particular cluster." '121 Patent at 13:4-7 (emphasis added). Again, this section does not specifically state that the "memory line state information" is referring to states in a cache coherence protocol—that is because one of skill in the art would already understand that the term "state" in a reference discussing cache coherency would refer to cache coherence protocol states. However, it would make no sense for a "***coherence*** directory" to be concerned with states other than *coherence states*. Moreover, one of skill in the art would expect that if the phrase "memory line state information" was referring to

cache coherence protocol states plus some other states, it would describe what those other states are—but no description of "states" other than cache coherence protocol states is provided.

36. Moreover, in describing Figure 7, the patent simply states that "***the coherence directory 701 includes state information 713***" and "[i]n some embodiments, the memory line states are modified, owned, shared, and invalid." '121 Patent at 13:55-59 (emphasis added). This section does not say "cache coherence protocol state information" because one of ordinary skill in the art would already understand that, in the context of cache coherency, the relevant states are cache coherent protocol states. Indeed, Figure 7 confirms that the "state information" discussed in this passage is referring to cache coherence protocol states, showing "Modified, "Owned," "Shared, and "Invalid" states in the "State" column of the "Coherence Directory."



Coherence Directory 701			
Memory Line 711	State 713	Dirty Data Owner 715	Occupancy Vector 717
Address 721	Invalid	N/A	N/A
Address 731	Invalid	N/A	N/A
Address 741	Shared	N/A	Clusters 1,3
Address 751	Shared	N/A	Clusters 1, 2, 3, 4
Address 761	Owned	Cluster 4	Cluster 2, 3, 4
Address 771	Owned	Cluster 2	Cluster 2, 4
Address 781	Modified	Cluster 2	N/A
Address 791	Modified	Cluster 3	N/A
...	...	...	...

'121 Patent, Fig. 7.

37. I understand that Petitioners in certain *inter partes reviews* of the '121 Patent have alleged that the term "states" encompasses the condition of mere presence of data in a cache. To

the extent that Intel's proposed construction is intended to embrace such an understanding of "states," I disagree that it is an appropriate construction of the term. I am aware of no cache coherence protocol whose states consist of mere presence and non presence. I also do not believe that any such protocol exists, and I do not believe that such a protocol would be practical in providing cache coherence. Additionally, as described above, I believe that the term "state" in the '121 Patent refers to cache coherency protocol states.

38. Furthermore, as explained below, the "state" of the '121 patent is concerned with the *states* of lines which *are* stored in the cache. By contrast, presence information merely indicates *whether* a line is stored in the cache. The teachings of the '121 Patent confirm that mere presence is not a state.

39. For example, the '121 Patent teaches that "because the cache coherence directory provides information about ***where [i.e., in which cluster] memory lines are cached as well as their states***, probes only need be directed toward the clusters in ***which the requested memory line is cached***" and "[***t]he state of a particular cach<u>ed</u> line*** will determine what type of probe is generated." '121 Patent at 19:36-43 (emphasis added). This passage confirms that the relevant "state" as used in the '121 Patent is a state of a line that is "cached," i.e. it is already known that the line is present in one of the caches, and the "state" provides *additional* information about "a particular cached line."

40. This is also confirmed because the '121 Patent discusses the cache coherence directory, in which state information is stored, as limited in size compared to the size of the memory space. '121 Pat. at 18:54-58 ("Given the size of the memory space associated with each cluster, it is not practical to have an entry in the cache coherence directory for each memory line. Rather, the directory is sized in relation to the amount of cache . . ."). Thus, "the coherence directory is an associative memory which associates the memory line addresses with their remote

cache locations."' '121 Pat. at 18:61-63; *see also* '121 Patent at 19:46-50 ("The associative nature of the cache coherence directory . . . "). An "associative" memory permits directly accessing the memory based on content, for example a cache tag, rather than merely providing an index into the directory. The "state" information and other information regarding a line are stored as fields in a row in the associative memory which is associated with that tag. '121 Pat. Fig. 7. However, the "state" information for a line is only stored if the directory's associative memory has a row corresponding to the cache tag (i.e. the line is present therefore there is a row, containing state information). If the line were not present in any of the caches, searching for its tag would simply result in a miss in the search of the directory's associative memory. Thus, the teachings of the '121 Patent regarding a cache coherence directory confirm that if a line is not cached, there is no state information about the line—i.e. in the '121 Patent presence in a cache is distinct from and a pre-condition to the existence of a "state" for that cache line.

41. The '121 Patent further expressly confirms that it does not use the term "state" to refer to mere presence because per-processing node presence information is stored in an "occupancy vector" in the '121 Patent. The '121 Patent explains that "[a]ny mechanism for tracking what clusters hold a copy of the relevant memory line in cache is referred to herein as an occupancy vector." '121 Patent at 14:2-4. It provides the example of the occupancy vector "implemented as an N-bit string, where each bit represents the availability of the data in the cache of N clusters." '121 Pat. at 13:67-14:2. Moreover, the '121 Patent distinguishes the "occupancy vector" from "state information." For example, in describing Figure 7, the '121 Patent states that "the coherence directory 701 ***includes state information 713***, dirty data owner information 715, and an ***occupancy vector 717*** associated with the memory lines 711." '121 Pat. at 13:55-57. Furthermore, Figure 7 itself further confirms that the mere presence information is not a state, depicting cache coherence protocol states—which are labeled "state information" as

distinct from presence information—labeled as the "occupancy vector." The '121 Patent would not introduce the term "Occupancy Vector" if "State" already included "presence." Furthermore, the '121 Patent would simply fail to operate if "state" was merely presence.

Coherence Directory 701			
Memory Line 711	State 713	Dirty Data Owner 715	Occupancy Vector 717
Address 721	Invalid	N/A	N/A
Address 731	Invalid	N/A	N/A
Address 741	Shared	N/A	Clusters 1,3
Address 751	Shared	N/A	Clusters 1, 2, 3, 4
Address 761	Owned	Cluster 4	Cluster 2, 3, 4
Address 771	Owned	Cluster 2	Cluster 2, 4
Address 781	Modified	Cluster 2	N/A
Address 791	Modified	Cluster 3	N/A
...	...	...	...

'121 Patent, Fig. 7[6]. Thus, the figure further confirms that presence information is distinct from the "state" of the "memory line," as the only "State[s]" shown in Fig. 7 are cache coherence protocol states.

42. Figure 8 of the '121 patent is also informative. Figure 8 shows how probe filter information can be used to reduce the number of nodes that need to be probed. Notably, in the column labeled "Probe Filter Information 821", the entries that are listed--"invalid", "shared", "owned," and "modified" (MOSI protocol)—exactly correspond with the entries in the "State" column of the coherence directory of Figure 7. This further confirms that the "probe filter

6 In the rows for Address 781, and 791, the column "Dirty Data Owner" instead provides the presence information. Because those lines are in the "Modified" state, only one cluster has a valid copy of the cache line, and a "vector" is not needed.

information representative of states associated with selected ones of the cache memories" of the claims refers to cache coherence protocol states and not to occupancy or presence information.

43. I understand that Intel cites a passage of the '121 Patent that includes the following discussion:

> Although the coherence directory 701 includes the four states of modified, owned, shared, and invalid [i.e., the MOSI protocol], it should be noted that particular implementations may use a different set of states. In one example, a system may have the five states of modified, exclusive, owned, shared, and invalid [i.e., MOESI protocol]. The techniques of the present invention can be used with a variety of different possible memory line states.

'121 Patent at 14:30-36; Joint Claim Chart Ex. A at 21 (citing'121 Patent at 13:54-14:36). This passage's statement that the invention can be used with a "variety" of different possible memory line states does not support that the invention can be used with states other than cache coherence protocol states. To the contrary, one of skill in the art would recognize the phrase "a variety of different possible memory line states" as meaning "a variety of different possible cache coherence protocol states" because the '121 Patent is deeply connected to the field of cache coherency. Thus, this passage merely states that the inventions of the '121 Patent are not limited to the states of any particular cache coherence protocol (e.g. MOSI, MOESI, etc.)—not that some "state" other than a cache coherence protocol state can be used to practice the invention.

44. Indeed, neither this passage, nor anywhere else in the '121 Patent, describes probe filtering based on probe filtering information representative of states other than cache coherence protocols states. Moreover, there is no reason to believe that such a configuration could work.

45. Moreover, the MOESI states are not the universe of all cache coherence protocol states. For example, the Dragon protocol, a well known cache coherence protocol, uses a very different set of states than the MOESI states. In the Dragon protocol, the states are Exclusive, Shared-Clean, Shared-Modified, and Dirty. D. Culler & J. Singh, *Parallel Computer Architecture, A Hardware/Software Approach* (1999) at 302. The Dragon protocol has no

"Invalid" state." Culler at 302. Thus, saying that "a variety of different possible memory line states" can be used with the present invention, after mentioning the MOESI protocol, is simply referring to the universe of different cache coherence protocols—not states which are not cache coherence protocol states.

46. I also understand that Intel cites to a passage that states:

> According to a specific embodiment, the directory of shared states may be implemented as described above with reference to FIGS. 7 and 8, and indicates where particular memory lines are cached within the cluster.

'121 Patent at 28:29-34; Joint Claim Chart, Ex. A (citing 28:29-35). This passage does not imply that "states" "indicates where particular memory lines are cached within the cluster." To the contrary, the phrase "indicates where . . ." is referring to the "directory" which is depicted in Figures 7 and 8. As Figures 7 and 8 depict, in addition to "states," the directory also stores "dirty data owner" information and an "occupancy vector"—and it is those items, not "states" which "indicate[] where particular memory lines are cached within the cluster." Interpreting this passage as teaching that "states" "indicate where particular memory lines are cached" would ignore the express teachings of the very figures that the passage is discussing, as well as the knowledge of the field demonstrated by the articles and treatises referenced above.

Dated: September 29, 2015

[illegible]

Mark Jones, Ph.D.